Contents

```
/* * Program:  Basic linked list creation */
#include<stdio.h>
#include<stdlib.h>

int main()
{
  //node structure
  struct node
  {
     int data;
     struct node *next;                                  //link for the address
  };

  struct node *head,*middle,*last, *temp;                //declaring nodes

  //allocating memory for each node
  head   = malloc(sizeof(struct node));
  middle = malloc(sizeof(struct node));
  last   = malloc(sizeof(struct node));

  //assigning values to each node
  head->data   = 45;
  middle->data = 98;
  last->data   = 3;

  //connecting each nodes head->middle->last
  head->next   = middle;
  middle->next = last;
  last->next   = NULL;

  //temp is a reference for head pointer.
  temp = head;

  //till the node becomes null, printing each nodes data
  while(temp != NULL)
  {
     printf("%d->",temp->data);
     temp = temp->next;
  }
  printf("NULL");

  return 0;
}
```

Output

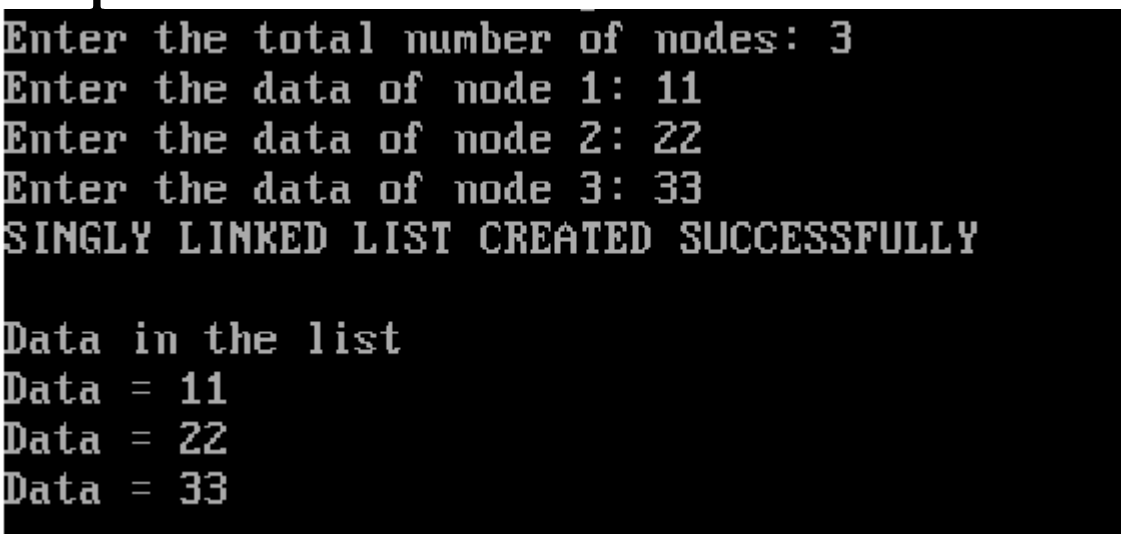
```
45->98->3->NULL
```

/* C program to create 'n' number of nodes in singly linked list and display them */**

```
#include <stdio.h>
#include <stdlib.h>

/* Structure of a node */
struct node {
int data;   // Data
struct node *next; // Address
}*head;

void createlist(int n);
void displaylist();

int main()
{
int n, data;
/** Create a singly linked list of n nodes*/
printf("Enter the total number of nodes: ");
scanf("%d", &n);
createlist(n);

printf("\nData in the list \n");
displaylist();

return 0;

}

/** Create a list of n nodes*/

void createlist(int n)
{
struct node *newNode, *temp;
int data, i;
head = (struct node *)malloc(sizeof(struct node));
/** Input data of node from the user*/
printf("Enter the data of node 1: ");
scanf("%d", &data);
head->data = data;                    // Link data field with data
head->next = NULL;                    // Link address field to NULL
temp = head;

/** Create n nodes and adds to linked list*/
for(i=2; i<=n; i++)
{
newNode = (struct node *)malloc(sizeof(struct node));
printf("Enter the data of node %d: ", i);
scanf("%d", &data);
newNode->data = data;                        // Link data field of newNode with data
newNode->next = NULL;                        // Link address field of newNode with NULL
```

```
temp->next = newNode;                    // Link previous node i.e.temp to the newNode

temp = temp->next;
}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
}

/** Display entire list*/
void displaylist()
{
struct node *temp;
/** If the list is empty i.e. head = NULL*/
if(head == NULL)
{
printf("List is empty");
}
else
{
temp = head;
while(temp != NULL)
{
printf("Data = %d\n", temp->data);          // Print data of current node
temp = temp->next;                          // Move to next node
}
}
}
```

Output



/* C program to insert a new node at the beginning of a Singly Linked List */**

```
#include <stdio.h>
#include <stdlib.h>

/* Structure of a node */
struct node {
int data;   // Data
struct node *next; // Address
}*head;

void createlist(int n);
void insertNodeAtBeginning(int data);
void displaylist();

int main()
{
int n, data;
/** Create a singly linked list of n nodes*/
printf("Enter the total number of nodes: ");
scanf("%d", &n);
createlist(n);

printf("\nData in the list \n");
displaylist();

/** Insert data at the beginning of the singly linked list*/
printf("\nEnter data to insert at beginning of the list: ");
scanf("%d", &data);

insertNodeAtBeginning(data);
printf("\nData in the list after insertion \n");

displaylist();
return 0;

}

/** Create a list of n nodes*/

void createlist(int n)
{
struct node *newNode, *temp;
int data, i;
head = (struct node *)malloc(sizeof(struct node));
/** Input data of node from the user*/
printf("Enter the data of node 1: ");
scanf("%d", &data);
head->data = data;                    // Link data field with data
head->next = NULL;                    // Link address field to NULL
temp = head;
```

```
/** Create n nodes and adds to linked list*/
for(i=2; i<=n; i++)
{
newNode = (struct node *)malloc(sizeof(struct node));
printf("Enter the data of node %d: ", i);
scanf("%d", &data);
newNode->data = data;                    // Link data field of newNode with data
newNode->next = NULL;                    // Link address field of newNode with NULL
temp->next = newNode;                    // Link previous node i.e.temp to the newNode

temp = temp->next;
}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
}

/** Create a new node and inserts at the beginning of the linked list.*/
void insertNodeAtBeginning(int data)
{
   struct node *newNode;
   newNode = (struct node*)malloc(sizeof(struct node));

   newNode->data = data;                                  // Link data part
   newNode->next = head;                                  // Link address part
   head = newNode;                                        // Make newNode as first node
   printf("DATA INSERTED SUCCESSFULLY\n");

}
/** Display entire list*/
void displaylist()
{
struct node *temp;
/** If the list is empty i.e. head = NULL*/
if(head == NULL)
{
printf("List is empty");
}
else
{
temp = head;
while(temp != NULL)
{
printf("Data = %d\n", temp->data);                     // Print data of current node
temp = temp->next;                                     // Move to next node
}
}
}
```

Output

```
Enter the total number of nodes: 4
Enter the data of node 1: 10
Enter the data of node 2: 20
Enter the data of node 3: 30
Enter the data of node 4: 40
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 10
Data = 20
Data = 30
Data = 40

Enter data to insert at beginning of the list: 90
DATA INSERTED SUCCESSFULLY

Data in the list after insertion
Data = 90
Data = 10
Data = 20
Data = 30
Data = 40
```

```
/*C program to insert new node at the middle of Singly Linked List*/

#include <stdio.h>
#include <stdlib.h>


/* Structure of a node */
struct node {
    int data;          // Data
    struct node *next; // Address
}*head;


void createlist(int n);
void insertNodeAtMiddle(int data, int position);
void displaylist();


int main()
{
    int n, data, position;

    /*Create a singly linked list of n nodes*/
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createlist(n);

    printf("\nData in the list \n");
    displaylist();

    /*Insert data at middle of the singly linked list*/
    printf("\n Enter data to insert at middle of the list: ");
    scanf("%d", &data);

    printf("\n Enter the position to insert new node: " );
    scanf("%d", &position);
    insertNodeAtMiddle(data, position);

    printf("\nData in the linked list  created  \n");
    displaylist();

    return 0;
}

/*Create a list of n nodes */
void createlist(int n)
{
    struct node *newNode, *temp;
    int data, i;

    head = (struct node *)malloc(sizeof(struct node));
```

```
        /*Input data of node from the user*/
        printf("Enter the data of node 1: ");
        scanf("%d", &data);

        head->data = data;       // Link the data field with data
        head->next = NULL;       // Link the address field to NULL

        temp = head;

        /*Creates n nodes and adds to linked list*/
        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));

                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data; //Link the data field of NewNode with data
                newNode->next = NULL; //Link the address field of newNode with NULL

                temp->next = newNode; //Link previous node i.e. temp to the newNode
                temp = temp->next;
         }

        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
  }


void insertNodeAtMiddle(int data, int position)
{
int i;
struct node *newNode, *temp;

newNode = (struct node*)malloc(sizeof(struct node));
newNode->data = data;            // data part
newNode->next = NULL;
temp = head;

/*Traverse to the n-1 position*/
for(i=2; i<=position-1; i++)
{
temp = temp->next;
if(temp == NULL)
break;
}
if(temp != NULL)
{
/* Link address part of new node */
newNode->next = temp->next;
/* Link address part of n-1 node */
```

```
temp->next = newNode;
printf("DATA INSERTED SUCCESSFULLY\n");
}
}
}

/*Display entire list*/
void displaylist()
{
    struct node *temp;

    /*If the list is empty i.e. head = NULL*/
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data); // Print data of current node
            temp = temp->next;                 // Move to next node
        }
    }
}
```

output

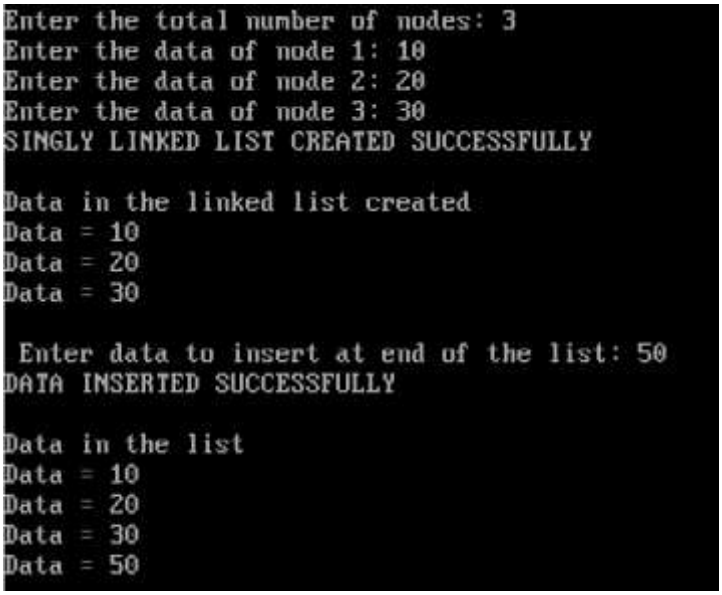
```
Enter the total number of nodes: 3
Enter the data of node 1: 10
Enter the data of node 2: 20
Enter the data of node 3: 30
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 10
Data = 20
Data = 30

 Enter data to insert at middle of the list: 60

 Enter the position to insert new node: 2
DATA INSERTED SUCCESSFULLY

Data in the linked list  created
Data = 10
Data = 60
Data = 20
Data = 30
```

```
/*C program to insert new node at the end of Singly Linked List*/

#include <stdio.h>
#include <stdlib.h>

/* Structure of a node */
struct node {
   int data;         // Data
   struct node *next; // Address
}*head;

void createlist(int n);
void insertnodeatEnd(int data);
void displaylist();

int main()
{
   int n, data, position;

   /*Create a singly linked list of n nodes*/
   printf("Enter the total number of nodes: ");
   scanf("%d", &n);
   createlist(n);

   printf("\nData in the list \n");
   displaylist();

   /*Insert data at middle of the singly linked list*/
   printf("\n Enter data to insert at end of the list: ");
   scanf("%d", &data);


   insertnodeatEnd(data);

   printf("\nData in the linked list created \n");
   displaylist();

   return 0;
}

/*Create a list of n nodes */
void createlist(int n)
{
   struct node *newNode, *temp;
   int data, i;
```

```
    head = (struct node *)malloc(sizeof(struct node));

    /*Input data of node from the user*/
    printf("Enter the data of node 1: ");
    scanf("%d", &data);


    head->data = data;                          // Link the data field with data
    head->next = NULL;                          // Link the address field to NULL

    temp = head;

    /*Creates n nodes and adds to linked list*/
    for(i=2; i<=n; i++)
        {
          newNode = (struct node *)malloc(sizeof(struct node));

             printf("Enter the data of node %d: ", i);
             scanf("%d", &data);

             newNode->data = data;              //Link the data field of NewNode with data
             newNode->next = NULL;              //Link the address field of newNode with NULL

             temp->next = newNode;              //Link previous node i.e. temp to the newNode
             temp = temp->next;

        }

        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");

}

/*Create a new node and inserts at the end of the linked list.
 */
void insertnodeatEnd(int data)
{
    struct node *newNode, *temp;

    newNode = (struct node*)malloc(sizeof(struct node));

    newNode->data = data;                                   // Link the data part
    newNode->next = NULL;

        temp = head;

        // Traverse to the last node
```

```
        while(temp->next != NULL)                    //check if it is not the last node
            temp = temp->next;

        temp->next = newNode;                        // Link address part

        printf("DATA INSERTED SUCCESSFULLY\n");

}

/*Display entire list*/
void displaylist()
{
    struct node *temp;

    /*If the list is empty i.e. head = NULL*/
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data);       // Print data of current node
            temp = temp->next;                       // Move to next node
        }
    }
}
```

Output


Enter the total number of nodes: 3
Enter the data of node 1: 10
Enter the data of node 2: 20
Enter the data of node 3: 30
SINGLY LINKED LIST CREATED SUCCESSFULLY
Data in the linked list created
Data = 10
Data = 20
Data = 30
Enter data to insert at end of the list: 50
DATA INSERTED SUCCESSFULLY
Data in the list
Data = 10
Data = 20
Data = 30
Data = 50

/*C program to a delete node at the middle of Singly Linked List*/

```
#include <stdio.h>
#include <stdlib.h>

/* Structure of a node */
struct node {
   int data;                          // Data
   struct node *next;                 // Address
}*head;


void createlist(int n);
void delete(int position);
void displaylist();


int main()
{
   int n, data, position;

   /*Create a singly linked list of n nodes*/
   printf("Enter the total number of nodes: ");
   scanf("%d", &n);

   createlist(n);

   printf("\nData in the linked list created \n");
   displaylist();


   printf("Enter the position to delete node: " );
   scanf("%d", &position);
   delete(position);

   printf("\nData in the list after deletion \n");
   displaylist();

   return 0;
}

/*Create a list of n nodes */
void createlist(int n)
{
   struct node *newNode, *temp;
```

```
    int data, i;

    head = (struct node *)malloc(sizeof(struct node));

    /*Input data of node from the user*/
    printf("Enter the data of node 1: ");
      scanf("%d", &data);

      head->data = data;                    // Link the data field with data
      head->next = NULL;                    // Link the address field to NULL

      temp = head;

       /*Creates n nodes and adds to linked list*/
       for(i=2; i<=n; i++)
       {
          newNode = (struct node *)malloc(sizeof(struct node));

          printf("Enter the data of node %d: ", i);
          scanf("%d", &data);

          newNode->data = data;         //Link the data field of NewNode with data
          newNode->next = NULL;         //Link the address field of newNode with NULL

          temp->next = newNode;         //Link previous node i.e. temp to the newNode
          temp = temp->next;

       }

       printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
 }

void delete (int pos)
{
    struct node* del,* temp = head;             // Creating a temporary
                                                // variable pointing to head
    int i;
    if (pos == 1)                       //if deleting first node/headnode
    {
       printf("\nElement deleted is : %d\n", temp->data);
       head = head->next;                       // Advancing the head pointer
       temp->next = NULL;
       free(temp);                              // Node is deleted
    }
    else
    {
```

```
        for (i = 1; i < pos - 1; i++)
         {
            temp = temp->next;
         }
   // Now temp pointer points to the previous node of the node to be deleted

       del= temp->next;                              // del pointer points to the node to be deleted
        temp->next = temp->next->next;
        printf("\n Element deleted is : %d\n", del->data);
        del->next = NULL;
        free(del);                                                  // Node is deleted
    }

    return;
}



/*Display entire list*/
void displaylist()
{
    struct node *temp;

    /*If the list is empty i.e. head = NULL*/
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data); // Print data of current node
            temp = temp->next;                 // Move to next node
        }
    }
}
```

/Creation of doubly linked list and displaying the list in both direction*/**

```
#include <stdio.h>
#include <stdlib.h>

struct node {
   int data;
   struct node * prevptr;
   struct node * nextptr;
}*stnode, *ennode;


void DLLcreation(int n);
void displayDLL();
void displayDLLreverse();

int main()
{
   int n;
   stnode = NULL;
   ennode = NULL;
   printf(" Input the number of nodes : ");
   scanf("%d", &n);

   DLLcreation(n);
   displayDLL();
   displayDLLreverse();
   return 0;
}

void DLLcreation(int n)
{
   int i, data;
   struct node *fnNode;

   if(n >= 1)
   {
      stnode = (struct node *)malloc(sizeof(struct node));

         printf(" Input data for node 1 : ");              // assigning data in the first node
         scanf("%d", &data);

         stnode->data = data;
         stnode->prevptr = NULL;
         stnode->nextptr = NULL;
         ennode = stnode;
// putting data for rest of the nodes
         for(i=2; i<=n; i++)
         {
            fnNode = (struct node *)malloc(sizeof(struct node));
```

```
            printf(" Input data for node %d : ", i);
            scanf("%d", &data);
            fnNode->data = data;
            fnNode->prevptr = ennode;    // new node is linking with the previous node
            fnNode->nextptr = NULL;

            ennode->nextptr = fnNode;   // previous node is linking with the new node
            ennode = fnNode;            // assign new node as last node


        }

    }
}
void displayDLL()
{
    struct node * tmp;
    int n = 1;
    if(stnode == NULL)
    {
        printf(" No data found in the List yet.");
    }
    else
    {
        tmp = stnode;
        printf("\n\n Data entered on the list are :\n");

        while(tmp != NULL)
        {
            printf(" node %d : %d\n", n, tmp->data);
            n++;
            tmp = tmp->nextptr;             // current pointer moves to the next node


        }
    }
}

void displayDLLreverse()
{
    struct node * tmp;
    int n = 1;
    if(ennode == NULL)
    {
        printf(" No data found in the List yet.");
    }
    else
    {
        tmp = ennode;
        printf("\n\n Data entered on the list reverse  :\n");
```

```
        while(tmp != NULL)
        {
           printf(" node %d : %d\n", n, tmp->data);
           n++;
           tmp = tmp->prevptr; // current pointer moves to the next node



        }
    }
}
```

Output